REMARKS

Claims 1-15, 18-32 and 35 are currently pending in the application. By this amendment, claims 1-6, 12, 13, 18-23, 29, 30 and 35 are amended for the Examiner's consideration. Support for the amendment(s) is provided in at least the paragraphs spanning pages 17 and 18 and pages 25 and 26 of the specification. No new matter is added. Reconsideration of the rejected claims in view of the above amendments and the following remarks is respectfully requested.

35 U.S.C. §103 Rejections

Claims 1-4, 7, 18-21, 24 and 35 were rejected under 35 U.S.C. §103(a) by U.S. Patent No. 5,999,729 to Tabloski, Jr. et al. (Tabloski) in view of U.S. Patent No. 6,105,119 to Kerr. Claims 5, 6, 8-13, 22, 23 and 25-30 were rejected under 35 U.S.C. §103(a) over Tabloski in view of Kerr in view of U.S. Patent No. 6,018,627 to Iyengar et al. (Iyengar). Claims 14, 15, 31 and 32 were rejected under 35 U.S.C. §103(a) over Tabloski in view of Kerr and further in view of U.S. Patent No. 6,083,276 to Davidson et al. (Davidson). These rejections are respectfully traversed.

Currently there is a significant lack of programmer tools for providing definitions to a Dialogue Manager for creating and using forms. In fact, there is little or no tool support for the definition syntax of managed dialogue forms and thus this is a job for the experts only. But, the invention solves this problem by generating form data files which define electronic forms using the field of visual programming technology. In the invention, the method and system includes building a program in a development environment to represent a data file, compiling the program in the development environment into a software executable and running the executable to generate the data file containing definition files which are interpreted for future application execution. The applications are not themselves executed at this stage, only definitions are provided in

order to interpret the forms and to thus develop the forms and provide the correct

information to the user.

In particular, referring to Figure 6, for example, and the specification at pages 17

and 18,

> an application generator, and not the application
> itself is created using the visual builder 650. The
> executable program 645 from the development process is
> run (step 660) in order to generate a set of forms definitions
> (form data files) 640. These are then presented to the
> runtime dialogue manager 650 for it to interpret.

But, when the program produced is executed (step 660) it does not perform the task of the

final application but merely creates the textual definition of the forms required by the

Dialogue Manager.

These features, however, are not shown in the combination of references, as

applied by the Examiner. In Tabloski, for example, a parallel program development and

processing system includes a parallel program development section and a parallel

program execution section. The parallel program development section allows a program

developer to develop programs for execution by a parallel computer system using a

predetermined set of components which can be selected by the program developer, using

a graphical user interface, and linked in a dataflow graph that represents the order of

operations to be performed by the program on the data to be processed. (See Abstract.)

As disclosed at col. 2, lines 32-42:

> After the program developer has developed the graph
> defining the program under development, the parallel
> program development section generates executable
> program code from the instances of the components and
> modules and their interconnections, for execution by the
> parallel computer. In the executable program code,

> instances of the components and modules selected by the
> program developer form executable objects which are
> <u>executed by the parallel computer under control of a run-</u>
> <u>time system that includes an execution control object</u>.
> (Emphasis added.)

Accordingly, this reference does not show compiling the program in the development

environment into a software executable and running the executable to generate the data

file containing definition files which are interpreted by a dialogue manager system.

Tabloski appears to teach that the executable code is simply sent to the parallel computer

to execute the code. The developer, in the instance of Tabloski, creates a graph which

defines the program under development.

Also, it is discussed at columns 5 and 6 that the parallel program development

section 21 of Tabloski includes a user interface 30, a module information repository 31, a

program object repository 32, a program composition module 33 and a compiler 34. The

user interface 30 allows the program developer to develop the parallel program

graphically. After the program developer has graphically developed the parallel program,

the composition module 33 retrieves the object instance information from the program

object repository and generates a program in a high-level language. The high-level

language program generated by the program composition module is <u>compiled</u> by the

compiler 34, during which a run-time library 36 is linked in a conventional manner, to

generate the executable program 23. Again, nowhere is it described that the executable

software will be run to generate data files to provide definitions of forms.

Applicants first submit that the Kerr reference and the Tabloski reference are of

such divergent arts that one of ordinary skill in the art would not have known or been

provided with any motivation to combine these references. By way of example, the

Tabloski reference is clearly directed to a system and method for developing computer

programs, as the title implies. On the other hand, Kerr is directed to a data transfer

circuitry and particularly DSP circuitry. Two unrelated technologies which neither would provide one of ordinary skill in the art any motivation to combine.

Second, in any event, the combination of references would not even achieve the claimed invention. Kerr shows an optimal way to deploy a "pool" of MIPS available in a computer system at any given time. The system distributes and/or re-allocates its collective computational resources to satisfy a broad range of functional requirements on-the-fly. In this disclosure, there is reference to the use of VSP. Kerr does show using headers with information provided therein. But, at column 21, it is disclosed that the header file in the DirectDSP API guarantees that a compiler will use a common data structure. This is not even remotely related to invention which is directed to development tools using graphical means.

As to the dependent claims, Applicant submits that many of these features are also not shown in the applied references. For example, claim 4 recites that, on running the executable, at least one compiled component outputs its respective data file information into the data file. This is now taught or suggested in the references, as submitted by the Examiner. For example, the Examiner is of the opinion that this feature is shown in Tabloski at col. 14, lines 30-40. This section only teaches that the program developer can accept the information entered or cancel the information enters. Or, the program developer can provide information similar to the information previously described in another file. This passage, however, does not show the compiled component output its data file information. As submitted previously, in any event, Tabloski shows executing the application.

Also, Tabloski, in Figure 3, only shows a program development window (See, cols. 13 and 14). Applicants do not interpret this development window to allow, upon execution, at least one compiled component to cause another component to output its respective data file information into the data file. Figure 3 only shows different icons, and does not refer to any control. But, the developer can provide information as how the

record in the data received at the input of the instance of the represented icon is partitioned among the processors. (See, col. 14, lines 41-45.)

Further, Applicants submit that Figure 4 is not representative of a form. Figure 4, instead, shows a dialog box. This dialog boxes has a number of fields, but the dialog is not a form as used in the context of the invention.

Applicants also submit that the remaining claims are dependent on distinguishable independent claims and should thus be in condition for allowance for the reasons set forth above.

Applicants thus respectfully request that the rejections over all the claims be withdrawn.

CONCLUSION

In view of the foregoing amendments and remarks, Applicants submit that all of the claims are patentably distinct from the prior art of record and are in condition for allowance. The Examiner is respectfully requested to pass the above application to issue. The Examiner is invited to contact the undersigned at the telephone number listed below, if needed. Applicants hereby make a written conditional petition for extension of time, if required. Please charge any deficiencies in fees and credit any overpayment of fees to Attorney's Deposit Account No. 09-0457.

Respectfully submitted,

Andrew M. Calderon
Registration No. 38,093

McGuireWoods, LLP
Suite 1800
1750 Tysons Blvd.
McLean, VA 22102
(703) 712-5426

\\COM\278340.1